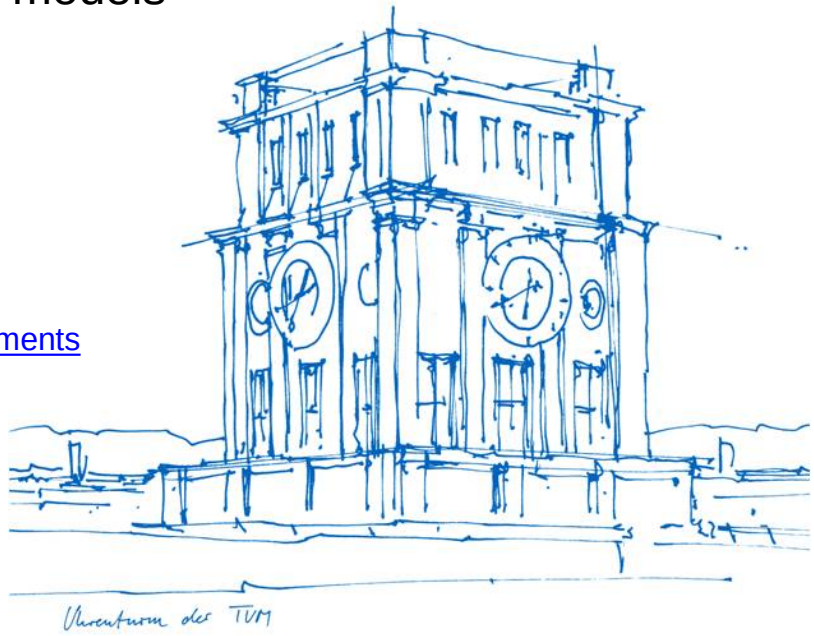
Machine Learning for Acoustics

- Supervised learning methods: discriminative models -

Marcus Maeder (Marcus.Maeder@tum.de)
Technical University of Munich
TUM School of Engineering and Design
Chair of Vibroacoustics of Vehicles and Machines

JOIN THE HACKATHON

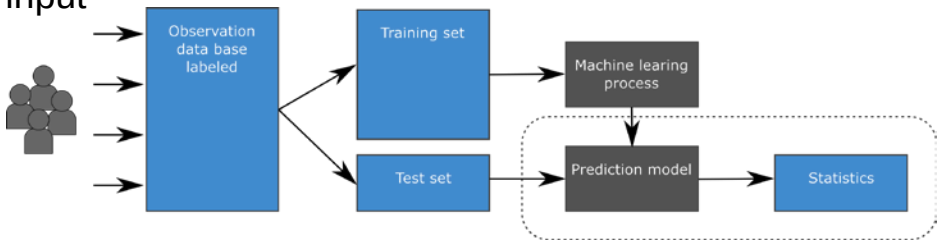
https://syncandshare.lrz.de/getlink/fi9UNhAZjJCihZ3AqFzwAB/Documents



Introduction – What are the differences

(Un)supervised, semi-supervised, reinforced learning

- Supervised Learning
  - Full set of labeled data is available when training the algorithm
  - Algorithms' task is to predict the given label after training
  - Useful for classification and regression problems
    - Classification: Predict discrete value wrt input data
    - Regression: Predict outcome of continuous input
  - Best suited when ground truth exists or set of reference points



Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Introduction – What are the differences

(Un)supervised, semi-supervised, reinforced learning

- Unsupervised learning
  - Data set that is given to the algorithm is unlabeled
  - Algorithm extracts common properties (features)
  - Organization of data
    - Clustering: Group training data w.r.t. similarities
    - Anomaly detection: Find outliers in dataset
    - Association: Find common features of data samples (Product recommendation)
    - Autoencoders: De-noising of data, e.g. pictures, video, and audio

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

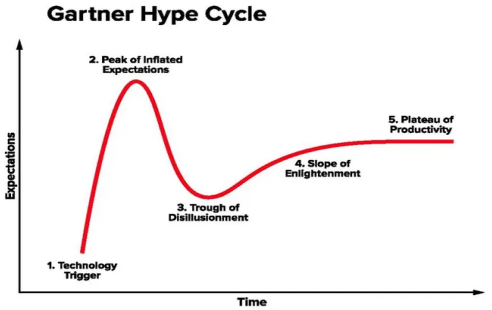
Introduction – What are the differences

(Un)supervised, semi-supervised, reinforced learning

- Semi-supervised learning
  - Mixture of labeled and unlabeled data as training set
  - Useful when extracting features is difficult and labeling is time-intensive
  - Use-cases
    - Analyzing medical pictures like CT or MRI scans
    - Creating general adversarial networks (GANs)
      - Two networks compete with each other (Generator vs. Discriminator)
      - Generator improves ability to trick the Discriminator
      - Discriminator improves ability to find fakes

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Introduction – What are the differences



(Un)supervised, semi-supervised, reinforced learning

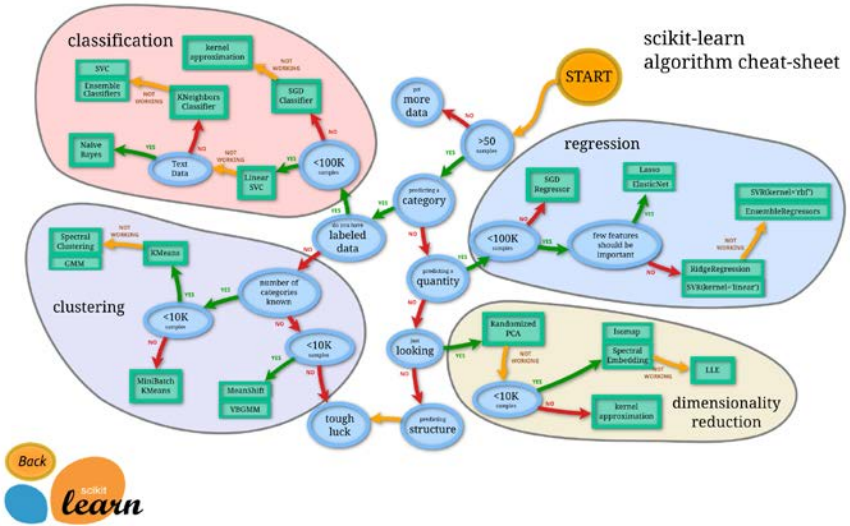
- Reinforced learning
  - The learning performance is enhanced by rewards and sanctions
  - AI agents find optimal way for accomplishing goal by predicting next best step via rewording
  - Relies on learning from past feedback and exploration of new tactics (long-term strategy)
  - Use-cases
    - Autonomous driving, managing warehouse inventory
    - Computer chip architecture, quantum computing
    - Real time translation

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Introduction – How to navigate

Algorithm selection of machine learning

- Understand the problem or task
- Specify
  - Task
  - Experience
  - Performance Measure
- Use predefined structures and algorithms
- Use the internet
  - Many problems have been solved already
  - Understand structures
  - Apply them to specific needs

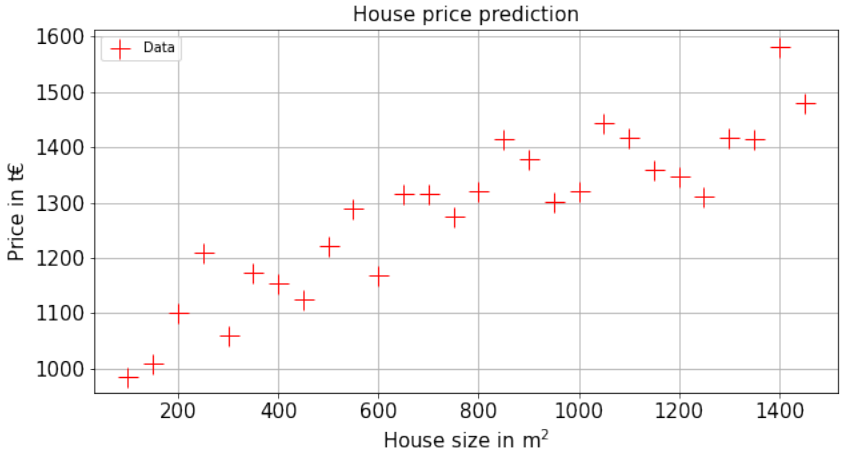


Source: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Introduction

Supervised learning

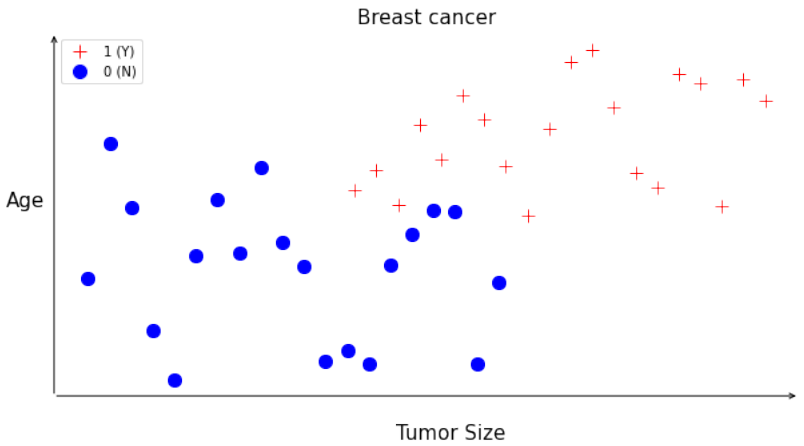
- Supervised Learning: Data is labeled or “right answer” is given
- Regression: Predict continuous values output



Introduction

Supervised learning

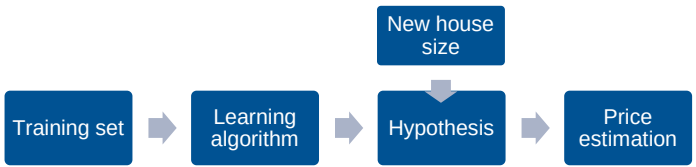
- Supervised Learning: Data is labeled or “right answer” is given
- Classification: Discrete valued output (0,1)
  - If we have more than one feature (Age, Tumor)
  - In real application number of features can be almost infinite (SVM are useful)
- In mechanical engineering
  - Health monitoring of machine
  - Quality identification using sound, vibration etc.

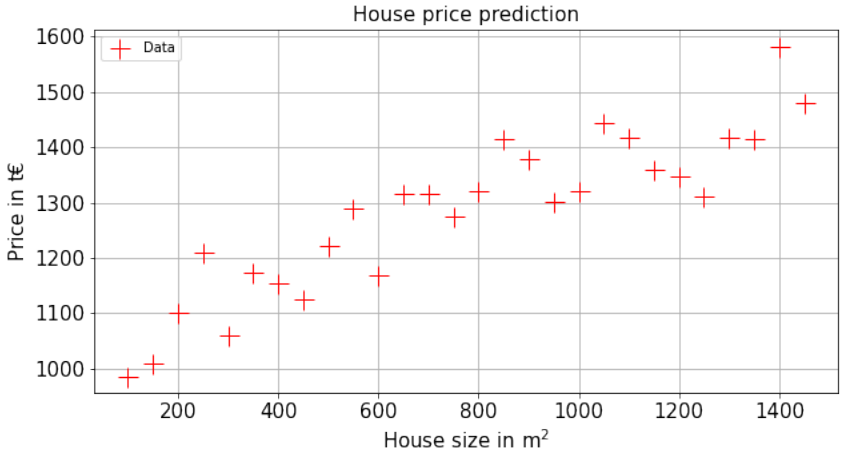


Supervised learning

Linear regression – Setting up the problem

- Let's take the housing prices as an example with the corresponding data set
- x… House size, y… House price
- The full data set is denoted by (x, y)
- With n samples of i combinations $(x^{(i)}, y^{(i)})$
- Task: Find the weights of the hypothesis $f(\Theta)$

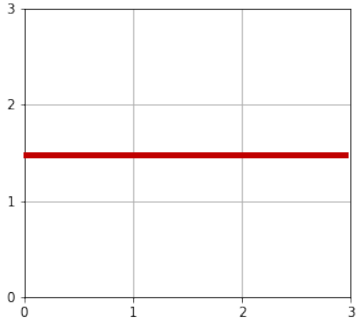


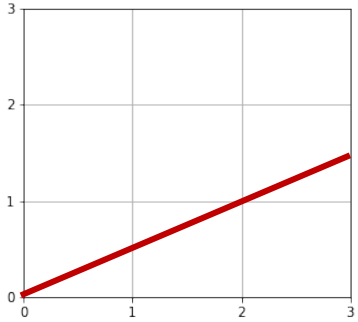


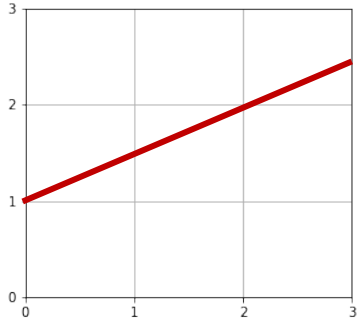
Supervised learning

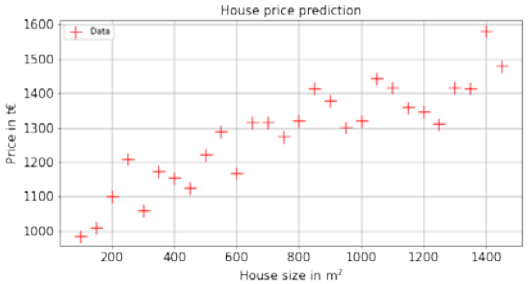
Linear regression – Finding the right hypothesis

- Let's say that there must be a linear relation in the data
- Therefore: $y = \theta_0 + \theta_1 x$ with $f(x, \theta_0, \theta_1) = f(x, \Theta)$
- Now let's try a few combinations $\Theta_1 = (1.5, 0)$, $\Theta_2 = (0, 0.5)$, $\Theta_3 = (1, 0.5)$









Supervised learning

Linear regression – The minimization problem to find parameters

- The minimization problem can be formulated as

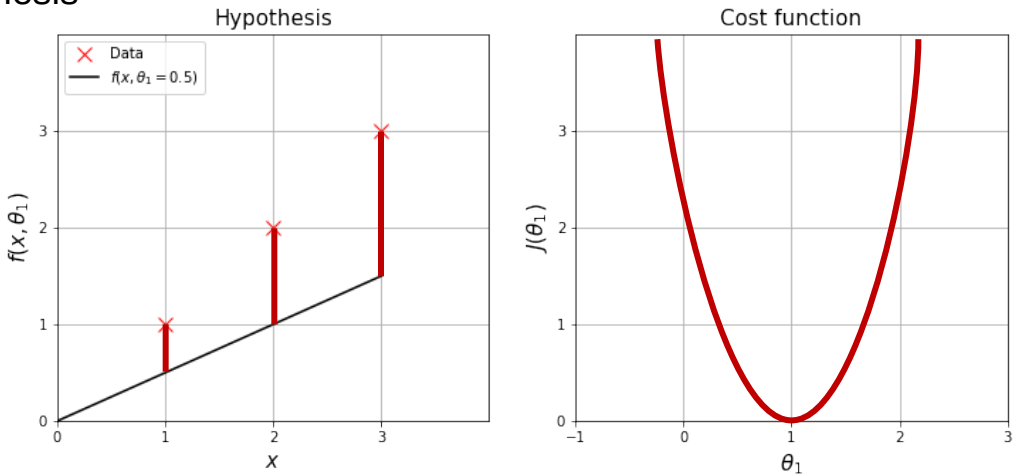
$$\min_{\theta_0,\theta_1} \frac{1}{2n} \sum_{i=1}^{n} \left(f\left(x^{(i)}, \theta_0, \theta_1\right) - y^{(i)} \right)^2$$

- Here $J(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(f\left(x^{(i)}, \theta_0, \theta_1\right) - y^{(i)} \right)^2$ is denoted as cost function which should be minimized
- Different cost functions can be chosen
- In linear regression this cost functions has proven to give good results

Supervised learning

Linear regression – Understanding the hypothesis and cost function

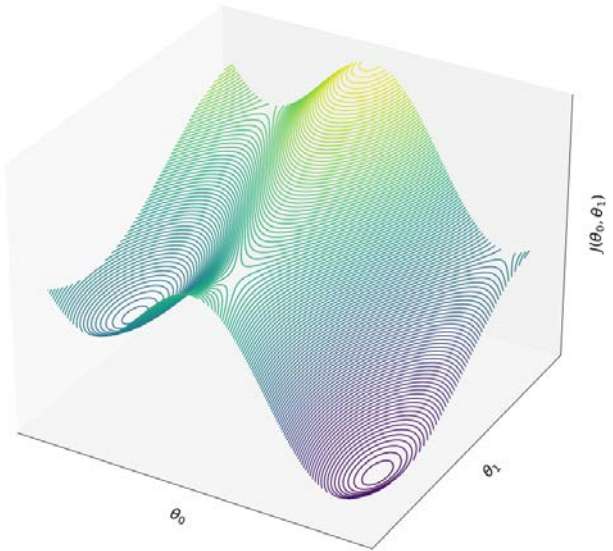
- Let's assume the data set $\{(1,1),(2,2),(3,3)\}$ and the hypothesis $f(x,\theta_1)=\theta_1 x$
- Find errors between data and hypothesis
- $J(\theta_1)=\frac{1}{2n}\sum_{i=1}^{n}\left(f\left(x^{(i)},\theta_1\right)-y^{(i)}\right)^2$
- Cost function minimizes at optimal value at $\theta_1=1$



Supervised learning

Gradient descent

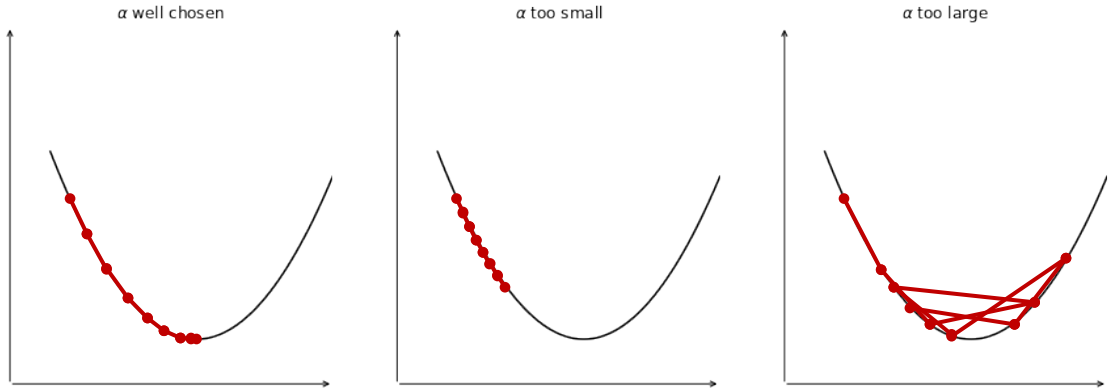
- The minimization problem:
  - $\min_{\theta_0,\theta_1} J(\theta_0,\theta_1) \rightarrow \min_{\theta_0,\theta_1,\dots\theta_n} J(\theta_0,\theta_1,\dots\theta_n)$
  - Set initial values ,e.g. $(\theta_0 = 0, \theta_1 = 0)$
  - Alter (θ_0,θ_1) until $J(\theta_0,\theta_1)$ is minimal
- Algorithm
  - Do until converged: $\{\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0,\theta_1)\}$ for $j = (0,1)$
  - Here α is known as learning rate (controls step size)
  - Be careful to compute all gradients before updating!!!



Supervised learning

Gradient descent – how it practically works

- The effect of setting the learning rate α:
- $\theta_1 \coloneqq \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J$
- Finds minimum since slope decreases close to minimum



Supervised learning

Gradient descent – applied to linear regression

- Gradient descent: $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\Theta)$
- Linear regression model: $J(\Theta) = \frac{1}{2n} \sum_{i=1}^{n} \left(f\left(x^{(i)}, \Theta\right) - y^{(i)} \right)^2$
- To make it work, we need: $\frac{\partial}{\partial \theta_j} J(\Theta)$
- Do some algebra and get: $\frac{\partial}{\partial \theta_j} J(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \left(f\left(x^{(i)}, \Theta\right) - y^{(i)} \right) \frac{\partial}{\partial \theta_j} f(\Theta)$
- For multiple features x_m : $f(\Theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m = \boldsymbol{\Theta}^{\mathrm{T}} \boldsymbol{x}$, where $x_0^{(i)} = 1$
- For linear regression: $\frac{\partial}{\partial \theta_j} J(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \left(f\left(x^{(i)}, \Theta\right) - y^{(i)} \right) x_j^{(i)}$

Supervised learning

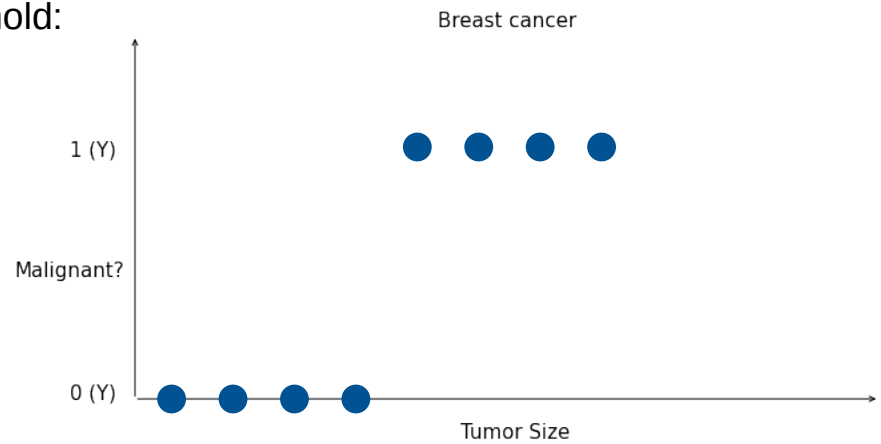
Gradient descent – helpful hints

- Reduce the number of iterations and make algorithm more robust
  - Feature scaling for elliptical contours: $x_i := \frac{x_i}{\max(x)}$
  - Mean normalization: $x_i := \frac{x_i - \mu_i}{s_i}$ μ_i ... Mean and s_i^2 ... Variance
  - Plot cost function $J(\Theta)$ vs. number of iterations; $J(\Theta)$ should decrease monotonically
  - Automatic convergence rate ε
- Debugging
  - If $J(\Theta)$ monotonically increases or oscillating; decrease learning rate α
  - Slow converging: increase learning rate α
  - Try a set of learning rates, such as $\alpha = 0.001 \cdot \frac{k+1}{3}$ with $k \in \mathbb{N}$, chose the one with best convergence

Supervised learning

Binary classification – Separating data into two categories

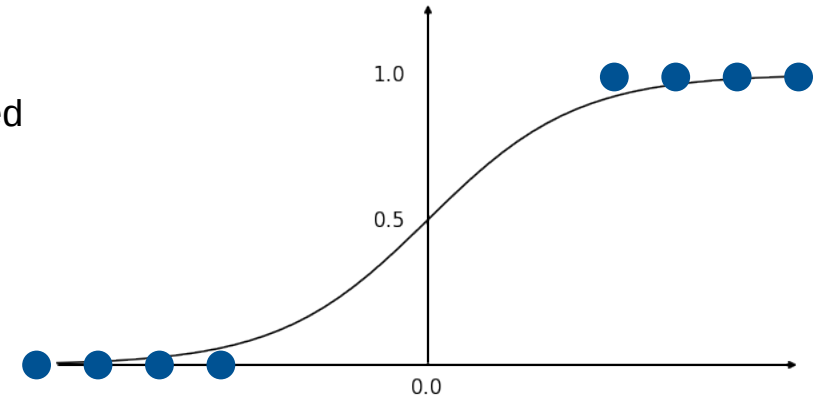
- In a set of data, the outcome takes only two values, i.e. $y \in \{0,1\}$ with 0: faulty, 1: ok or 0: absent and 1: present (arbitrary chosen)
- Using linear regression can work using threshold:
  - If $f(x,\theta) \geq 0.5 \Rightarrow y = 1$
  - If $f(x,\theta) < 0.5 \Rightarrow y = 0$
  - With additional data, the prediction can be prone to outliers
- Linear regression rarely works for classification



Supervised learning

Binary classification – Logistic Regression as classification algorithm

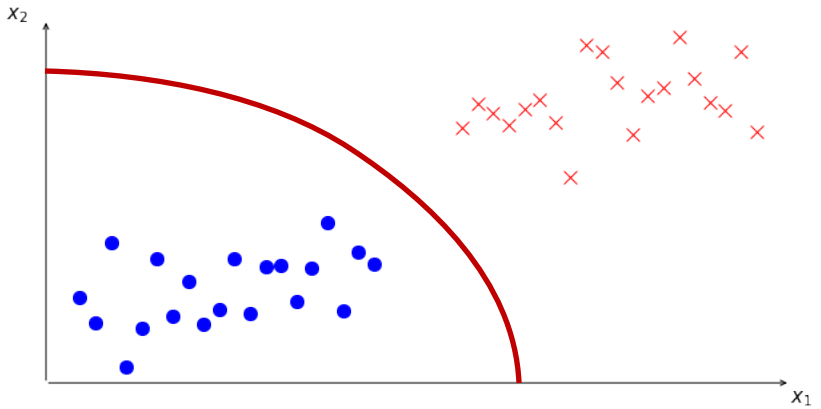
- Task: find an algorithm, where $0 \leq f(x,\theta) \leq 1$ to achieve binary classification
- Remember in linear regression: $f(x,\theta) = \boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{x}$
- In logistic regression $f(x,\theta) = g\left(\boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{x}\right)$
  - Sigmoid/Logistic function $g(z) = \frac{1}{1+e^{-z}}$
- Interpretation of logistic regression as the estimated probability that $\mathrm{y} = 1$ with respect to input $\boldsymbol{x}$
  - That means $f(x,\theta) = P(y = 1|\boldsymbol{x};\boldsymbol{\Theta})$
  - The probability for $\mathrm{y} = 1$, given $\boldsymbol{x}$, parameterized by $\boldsymbol{\Theta}$
  - $P(y = 0|\boldsymbol{x};\boldsymbol{\Theta}) = 1 - P(y = 1|\boldsymbol{x};\boldsymbol{\Theta})$



Supervised learning

Binary classification – Logistic Regression as classification algorithm

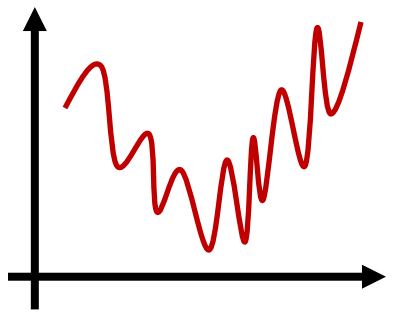
- Decision boundary for logistic regression $f(x,\theta) = g\left(\boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{x}\right)$ with $g(z) = \frac{1}{1+e^{-z}}$
  - Prediction that $\mathrm{y} = 1$ if $f(x,\theta) \geq 0.5$ i.e. $\boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{x} \geq 0$
  - Prediction that $\mathrm{y} = 0$ if $f(x,\theta) < 0.5$ i.e. $\boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{x} < 0$
  - Decision boundary where $f(x,\theta) = 0.5$
  - Parameters $\boldsymbol{\Theta}$ can be estimated using the data
  - Non-linear decision boundaries using higher order polynomial hypothesis, e.g.
    $f(x,\theta) = g\left(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \cdots\right)$



Supervised learning

Binary classification – How to get parameters for logistic regression

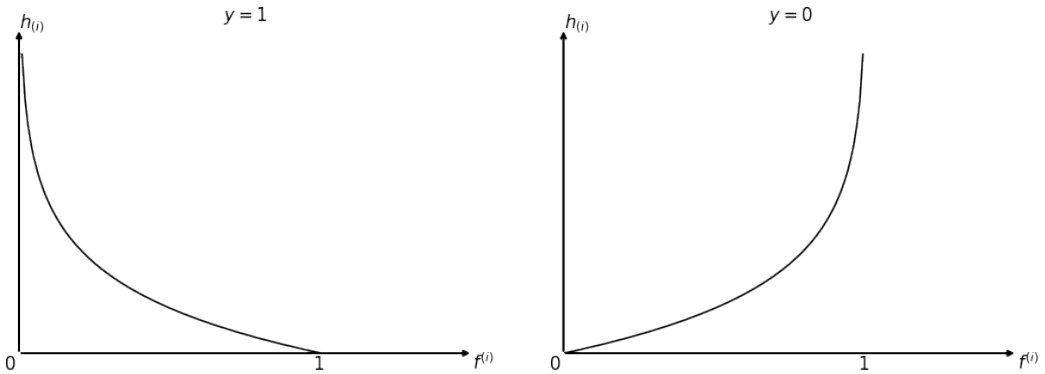
- Cost function for logistic regression
  - Using the squared cost function of linear regression will result in a non-convex behavior for logistic regression and therefore a different cost function is needed
  - Linear regression model: $J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(f\left(x^{(i)},\Theta\right) - y^{(i)}\right)^2 = \frac{1}{n}\sum_{i=1}^{n} h_i\left(f^{(i)}, y^{(i)}\right)$
  - Logistic regression model: $h_i\left(f^{(i)}, y^{(i)}\right) = \begin{cases} -\log\left(f^{(i)}\right), & y^{(i)} = 1 \\ -\log\left(1 - f^{(i)}\right), & y^{(i)} = 0 \end{cases}$
  - What does that mean for the error estimation?



Supervised learning

Binary classification – How to get parameters for logistic regression

- Cost function for logistic regression model: $h_i\left(f^{(i)}, y^{(i)}\right) = \begin{cases} -\log\left(f^{(i)}\right), & y^{(i)} = 1 \\ -\log\left(1 - f^{(i)}\right), & y^{(i)} = 0 \end{cases}$
- Wrong predictions are penalized with a very high cost h_i
- This cost function is convex and free of local minima



Supervised learning

Binary classification – How to get parameters for logistic regression

- To make it more convenient:

$$h_i\left(f^{(i)}, y^{(i)}\right) = -y^{(i)} \log\left(f^{(i)}\right) - \left(1 - y^{(i)}\right) \log\left(1 - f^{(i)}\right) \text{ for } y^{(i)} = \{0,1\}$$

- The cost functions for gradient descent:

$$J(\Theta) = \frac{1}{n} \sum_{i=1}^{n} h_i\left(f^{(i)}(\Theta), y^{(i)}\right) = -\frac{1}{n} \sum_{i=1}^{n} y^{(i)} \log\left(f^{(i)}(\Theta)\right) + \left(1 - y^{(i)}\right) \log\left(1 - f^{(i)}(\Theta)\right)$$

- Using gradient descent to fit parameters for Θ

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\Theta) \quad \text{with} \quad \frac{\partial}{\partial \theta_j} J(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \left(f\left(x^{(i)}, \Theta\right) - y^{(i)}\right) f'|_{x_j^{(i)}} x_j^{(i)}$$

- Remember that feature scaling can improve gradient descent
- Track the cost function $J(\Theta)$ for each iteration to check convergence

Supervised learning

Binary classification – To improve the parameter estimation

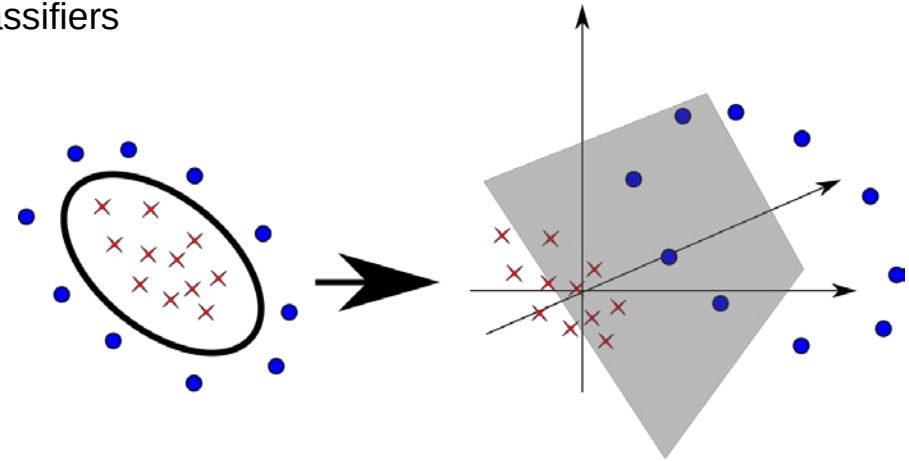
- There are alternative optimization algorithms
  - Conjugated gradient/BFGS/L-BFGS
  - Advantages
    - Picking α is not necessary
    - Faster than gradient descent
  - Disadvantages
    - Complex algorithms

Source: https://en.wikipedia.org/wiki/Limited-memory_BFGS

Support vector machines (SVM)

The idea behind

- Support Vector Machine (SVM) is a state-of-the-art classification method introduced in 1992 by Boser, Guyon, and Vapnik
- SVMs belong to the class of generalized linear classifiers
- Non-linear transformation transforms the non-linear separable data into a space in which they are linearly separable using hyperplanes
- Support vectors are determined (data sets near the decision boundary), which define an optimal decision boundary



Source: http://pyml.sourceforge.net/doc/howto.pdf

Support vector machines (SVM)

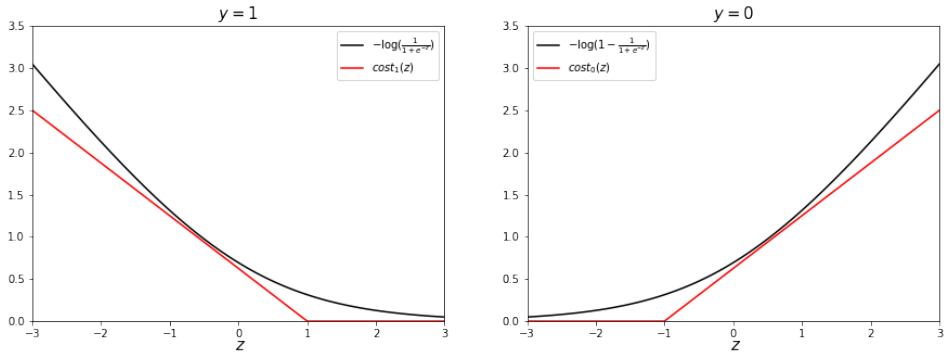
Applications

- Linear and nonlinear classification
- Linear and nonlinear regression
- One-class SVM, e.g. for outlier detection
- Easily configurable, at least when one of the standard kernels is used.
- Especially suitable in the case of very high dimensional data:
  - Object detection in image and video data
  - Document and text classification
  - Bioinformatics: sequencing of DNA and proteins

Support vector machines (SVM)

Optimization problem

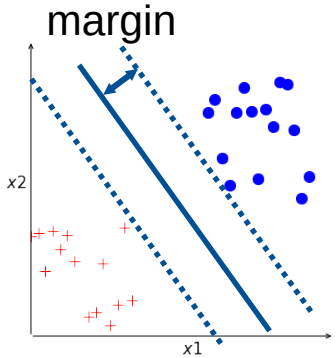
- Logistic regression: $J(\Theta) = \frac{1}{n}\left[\sum_{i=1}^{n} -y^{(i)} \log\left(f^{(i)}(\Theta)\right) - \left(1-y^{(i)}\right)\log\left(1-f^{(i)}(\Theta)\right) + \frac{\lambda}{2}\sum_{j=1}^{k}\theta_j^2\right]$
- Support vector machine $J_{SVM}(\Theta) = \frac{1}{n}\left[\sum_{i=1}^{n} y^{(i)}\,\mathrm{cost}_1\left(\Theta^{\mathrm{T}}\boldsymbol{x}\right) + \left(1-y^{(i)}\right)\mathrm{cost}_0\left(\Theta^{\mathrm{T}}\boldsymbol{x}\right) + \frac{\lambda}{2}\sum_{j=1}^{k}\theta_j^2\right]$
- Moreover the term $\frac{1}{n}$ is neglected and
- $J_{SVM}(\Theta) = A + \lambda B$ becomes $J_{SVM}(\Theta) = CA + B$ with $C = \frac{1}{\lambda}$
- Note that
  - $y = 1$ for $\Theta^{\mathrm{T}}\boldsymbol{x} \geq 1$
  - $y = 0$ for $\Theta^{\mathrm{T}}\boldsymbol{x} \leq -1$



Support vector machines (SVM)

Optimization problem

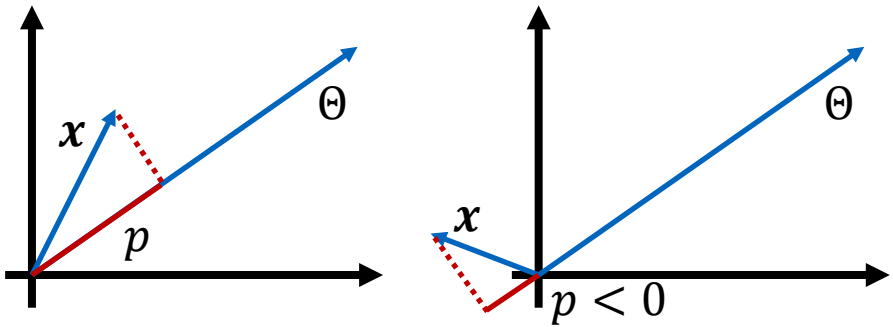
- For seeking $\Theta^{\mathrm{T}}\boldsymbol{x} \geq 1, y = 1$ or $\Theta^{\mathrm{T}}\boldsymbol{x} \leq -1, y = 0$ results in a safety margin
- Support vector machine $J_{SVM}(\Theta) = C\left[\sum_{i=1}^{n} y^{(i)}\,\mathrm{cost}_1\left(\Theta^{\mathrm{T}}\boldsymbol{x}\right) + \left(1 - y^{(i)}\right)\mathrm{cost}_0\left(\Theta^{\mathrm{T}}\boldsymbol{x}\right)\right] + \frac{1}{2}\sum_{j=1}^{k}\theta_j^2$
- If C is a very large value, i.e. $C \gg 1$ then $\min_{\Theta} J_{SVM}(\Theta) = \min_{\Theta} C \cdot 0 + \frac{1}{2}\sum_{j=1}^{k}\theta_j^2$
- This way SVM gets an optimal decision boundary with a margin
  - Margin gets maximized w.r.t. to decision boundary
  - Large margin classifier
  - If C is a very large, then SVM is sensitive to outliers



Support vector machines (SVM)

Mathematics behind SVM decision boundary

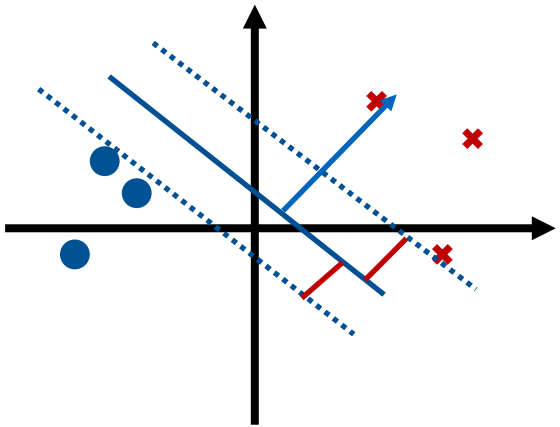
- $\min_{\Theta} C \cdot 0 + \frac{1}{2}\sum_{j=1}^{2} \theta_j^2 = \min_{\Theta} \frac{1}{2}\left(\theta_1^2 + \theta_2^2\right) = \min_{\Theta} \frac{1}{2}\left(\sqrt{\theta_1^2 + \theta_2^2}\right)^2 = \min_{\Theta} \frac{1}{2}\|\Theta\|^2$ subject to
  - For $y = 1$ then $\Theta^{\mathrm{T}}\boldsymbol{x} = p\|\Theta\| \geq 1$
  - For $y = 0$ then $\Theta^{\mathrm{T}}\boldsymbol{x} = p\|\Theta\| \leq -1$

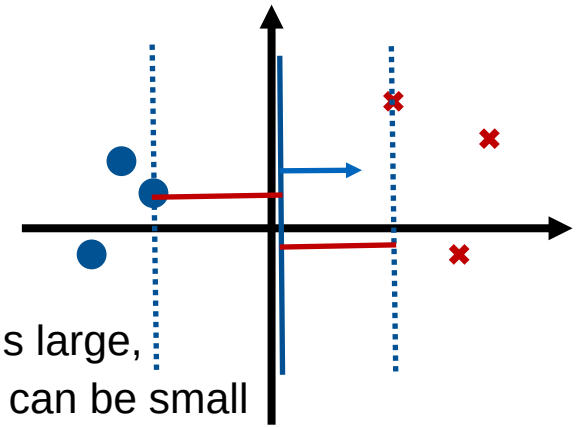


Support vector machines (SVM)

Mathematics behind SVM decision boundary

- $\min_{\Theta} \frac{1}{2}\|\Theta\|^2$ with $\Theta^{\mathrm{T}}\boldsymbol{x} = p^{(i)}\|\Theta\| \geq 1$ for $y^{(i)} = 1$ and $\Theta^{\mathrm{T}}\boldsymbol{x} = p^{(i)}\|\Theta\| \leq -1$ for $y^{(i)} = 0$
  where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector Θ



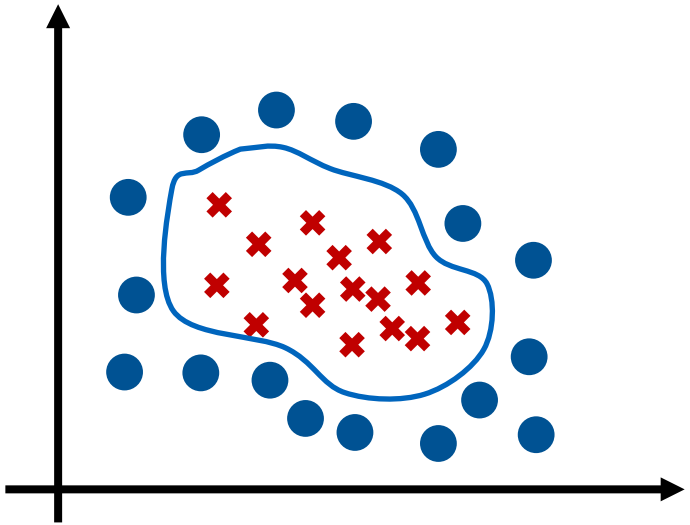


If p is large,
$\|\Theta\|$ can be small

Support vector machines (SVM)

Non-linear decision boundary

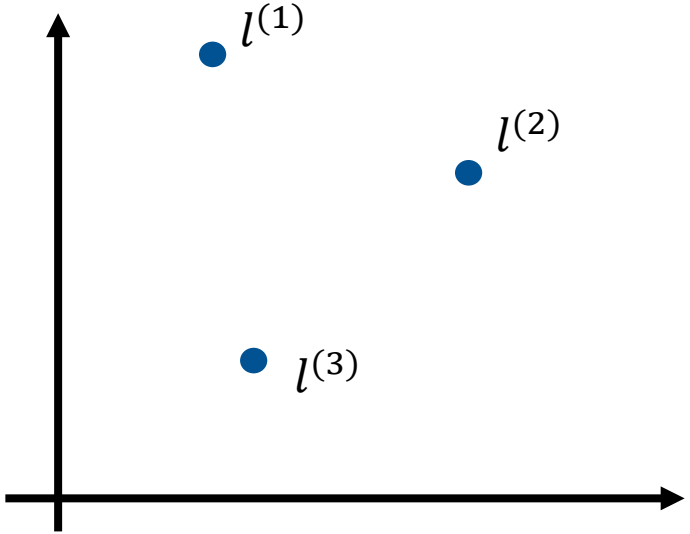
- Predict $y = 1$ if $\Theta^{\mathrm{T}}\boldsymbol{x} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \cdots \geq 1$
  - Features are: $\mathrm{f}_1 = x_1, \mathrm{f}_2 = x_2, \mathrm{f}_3 = x_1 x_2, \cdots$
  - Can we choose features in a better way?



Support vector machines (SVM)

Non-linear decision boundary

- Define landmarks $l^{(1)}, l^{(2)}, l^{(3)}$
- With given x, new features can be computed
  - $f_1 = \mathrm{k}\,(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$
  - $f_2 = \mathrm{k}\,(x, l^{(2)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$
  - $f_3 = \mathrm{k}\,(x, l^{(3)}) = \exp\left(-\frac{\|x - l^{(3)}\|^2}{2\sigma^2}\right)$



Support vector machines (SVM)

Interpretation of kernels

- $f_1 = \mathrm{k}\left(x, l^{(1)}\right) = \exp\left(-\frac{\left\|x - l^{(1)}\right\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{m}\left(x_j - l_j^{(1)}\right)^2}{2\sigma^2}\right)$
- If $x \approx l^{(1)}$

$$f_1 \approx \exp\left(-\frac{0}{2\sigma^2}\right) = 1$$

- If x is far away from $l^{(1)}$

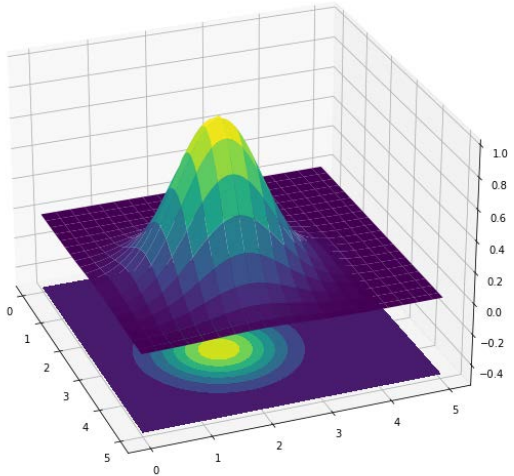
$$f_1 \approx \exp\left(-\frac{(large\ number)^2}{2\sigma^2}\right) = 0$$

- This way the landmarks can be seen as new features

Support vector machines (SVM)

Interpretation of kernels

- As an example $f_1 = \exp\left(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2}\right)$ with $l^{(1)} = \begin{bmatrix}3\\2\end{bmatrix}$ and $\sigma = 0.5$
- If data is close to landmark it will result in $f_1 = 1$ and far away $f_1 = 0$



Support vector machines (SVM)

SMV parameters

- $C = \frac{1}{\lambda}$
  - Large C, small λ: Lower bias, high variance
  - Small C, large λ: Higher bias, low variance
- σ^2
  - Large σ^2: Features $f^{(i)}$ vary more smoothly, higher bias, lower variance
  - Small σ^2: Features $f^{(i)}$ vary less smoothly, lower bias, higher variance

Support vector machines (SVM)

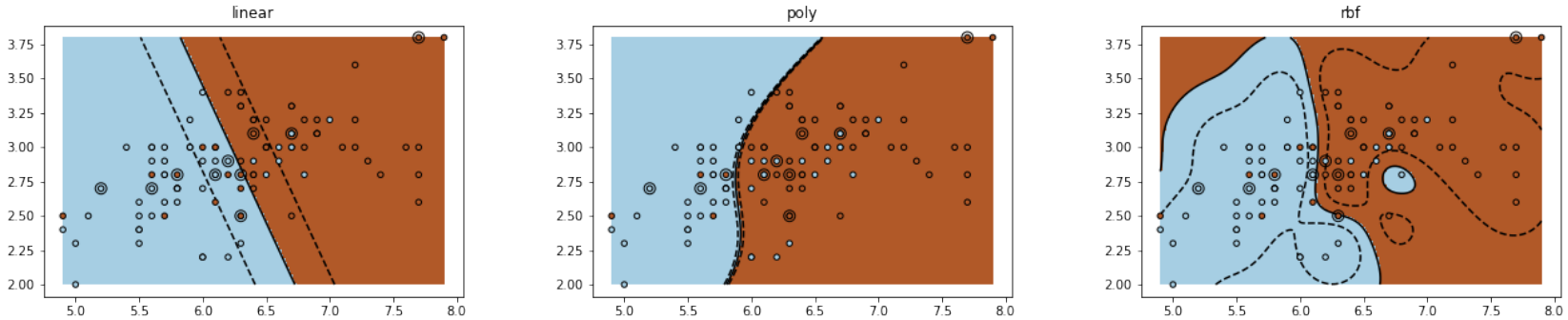
Practical notes when using SMV

- You need to specify C and the kernel
  - No kernel -> “linear kernel”; predict $y = 1$ if $\Theta^{\mathrm{T}}\boldsymbol{x} \geq 1$
    - Large number of features, small number of training set
  - Gaussian kernel $f_i = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$, where $l^{(i)} = x^{(i)}$ and choose σ^2
    - Small number of features, large number of training data
    - Sometimes the kernel function must be defined manually to map $\boldsymbol{x} \rightarrow \boldsymbol{f}$
    - Using feature scaling must be conducted before using Gaussian kernel

Support vector machines (SVM)

Analyzing a dataset with different SMV kernels

- Iris Dataset

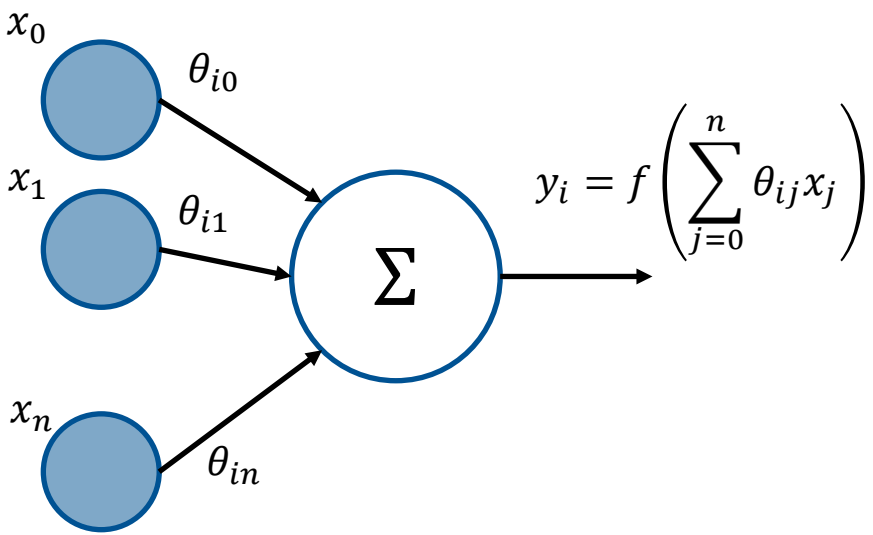


Source: https://scikit-learn.org/stable/modules/svm.html#svm-classification

Mathematical model of a single neuron

Mimicking the natural behavior of a neuron

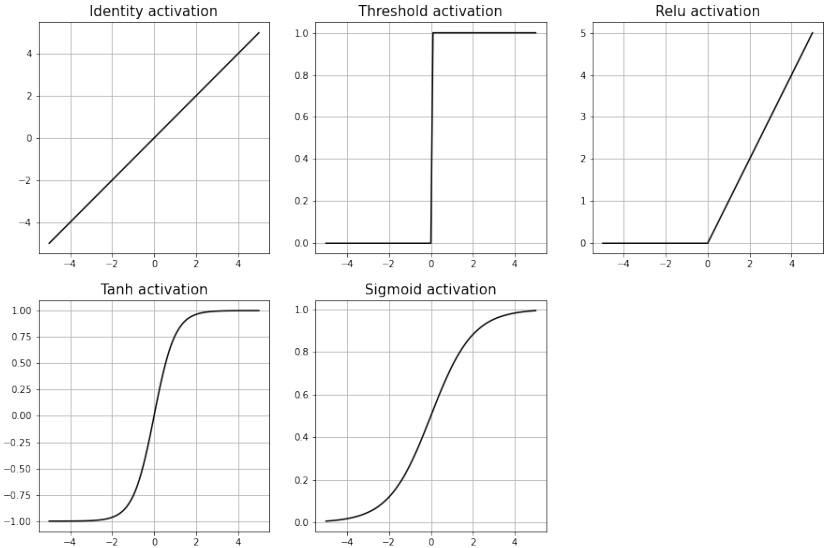
- Weights θ_{ij} represent connection strengths between neuron j in previous layer and neuron i in current layer.
- The activation function $f(z)$ forms the weighted sum of $z = \sum_{j=0}^{n} \theta_{ij} x_j$ of the input of the neuron into an output value.
- Activation function is chosen depending on the task (classification, regression), the learning algorithm, and the network topology.



Mathematical model of a single neuron

Different activation functions

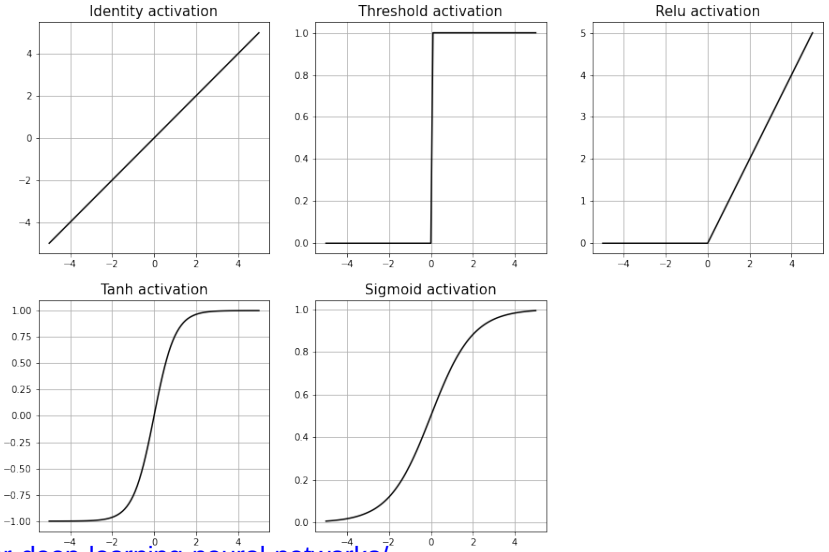
- Identity $f(x) = x$
  - Regression problems
- Threshold $f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$
  - Not continuous differentiable
- ReLU $f(x) = \max(0, x)$
  - Exit of convolution layer in convolutional neural network
  - Not continuous differentiable
    - Set value of derivative at $x = 0$, e.g. $(0,0.5,1)$
    - Softplus function $f(x) = \ln(1 + e^x)$



Mathematical model of a single neuron

Different activation functions

- Tanh $f(x) = \tanh x$
  - Nowadays preferred over sigmoid
  - Results in better training and better prediction
- Sigmoid $f(x) = \frac{1}{1+e^{-cx}}$
  - Classification problems $K = 2$
- Both functions saturate and are only sensitive around $x = 0$
- Very deep networks suffer from vanishing gradient problem (then use ReLU)

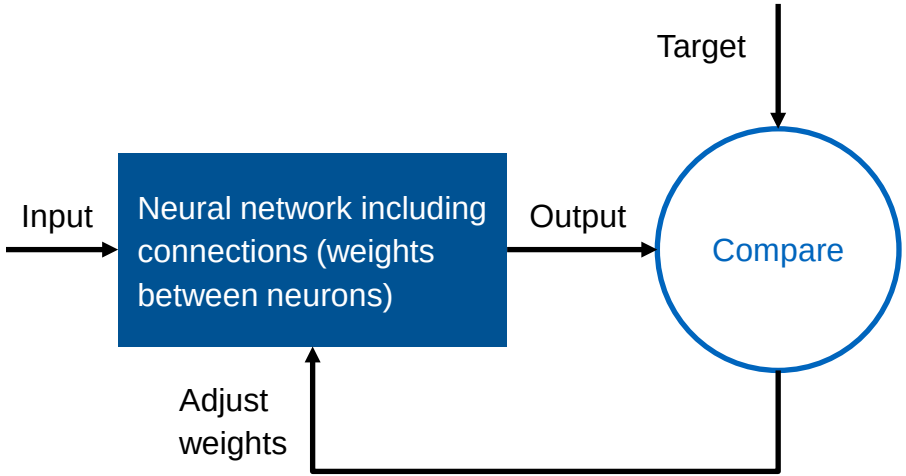


Source: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Mathematical model of a single neuron

Principle of artificial neural network

- In each iteration, the result (output) is compared with performance standard (target).
- Depending on this comparison, the weights of the connections in the neural network are adjusted.
- This adaptation represents the learning



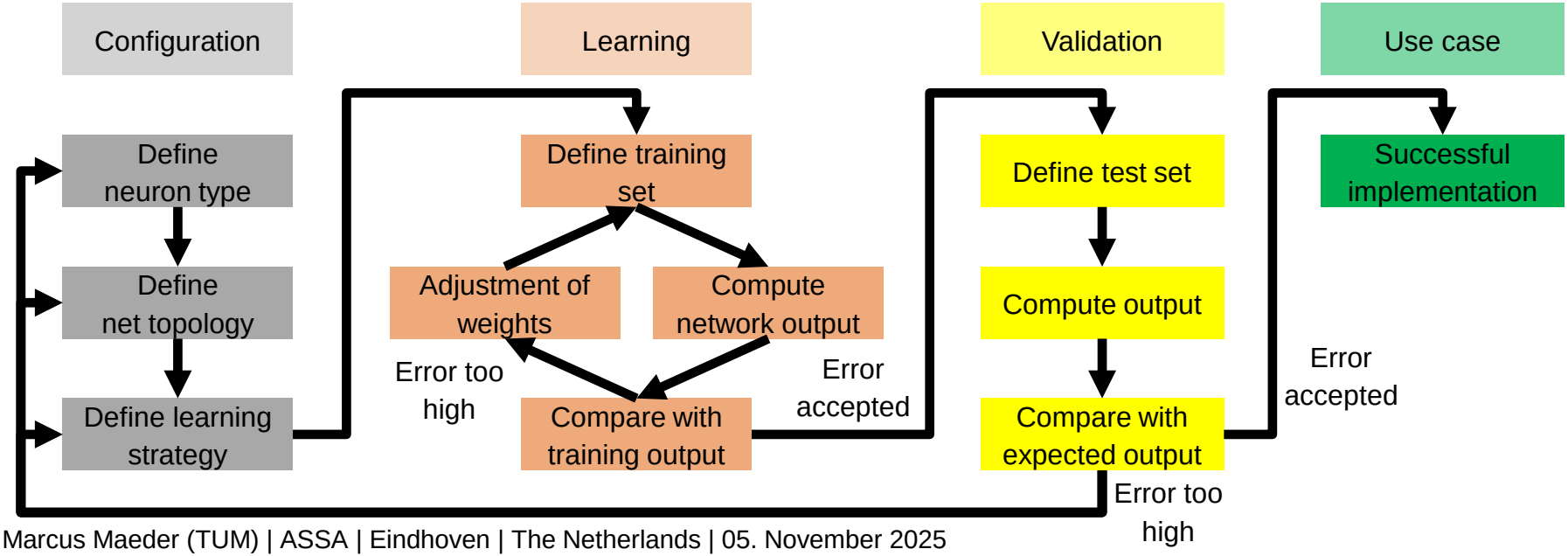
Categories of neural networks

Classification according to input and output

- **Classifiers**: Supervised learning with pairs of (input/associated class index)
- **Regression**: Supervised learning with pairs of (input/associated class index)
- **Generation**: of new data, e.g. with Generative Adversarial Networks (GANs) or Variational Autoencoders
- **Associators** : Similar input vectors (data sets) are grouped in clusters. The training data contain only inputs without the associated class (unsupervised learning)
- **DIM-Reduction**: N-dimensional space at the input of the network (represented by N input neurons) is transformed into a K-dimensional space $(K < N)$ at the output of the network (represented by K output neurons) (unsupervised).

Neural network design

The different phases



Single Perceptron

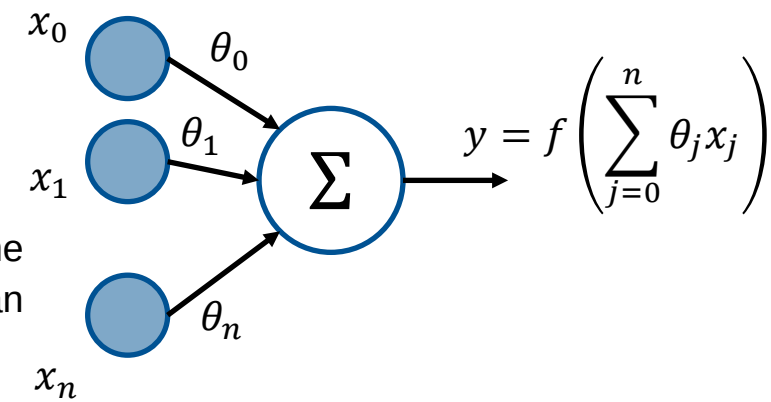
Definition

- **Single Layer Perceptron:** A neural feedforward network in which there are only input and output neurons and all neurons of the input layer are fully connected to the neurons of the output layer is called a single layer perceptron (SLP).
  - For Boolean classifiers ($K = 2$) usually one output neuron is enough
  - For ($K > 2$) classification, K output neurons are usually used
  - For regression with scalar function value, one output neuron is enough.

Single Perceptron

Perceptron with single output

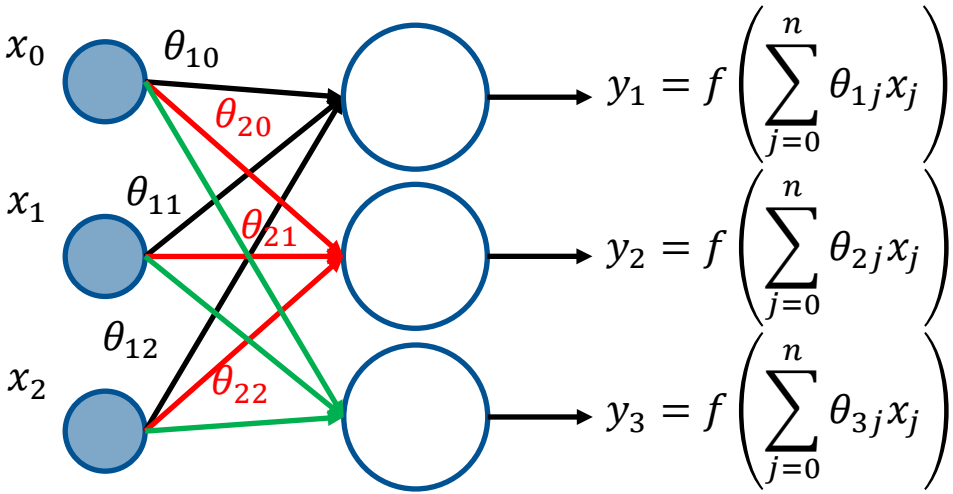
- Feature vector $(x_1, \dots, x_n)$
- Bias is usually constant $x_0 = 0$
- Weighted sum at the entrance of the perceptron
  - $\sum_{j=0}^{n} \theta_j x_j = \boldsymbol{\Theta}^{\mathbf{T}} \cdot \boldsymbol{x}$
- Bias $\theta_0 x_0$ often written as b
  - $y = f\left(\sum_{j=1}^{n} \theta_j x_j + b\right)$
- In classification where there are two outputs the threshold activation or sigmoid activation function can be used



Single Perceptron

SLP for classification within more than two classes

- The single-layer perceptron contains exactly K neurons in the output layer
- For each neuron there is one row in the weight matrix
- $\Theta = \begin{pmatrix} \theta_{10} & \theta_{11} & \theta_{12} \\ \theta_{20} & \theta_{21} & \theta_{22} \\ \theta_{30} & \theta_{31} & \theta_{32} \end{pmatrix}$



Single Perceptron

SLP for classification within more than two classes

- So far, we considered single layer perceptron
  - Contains only input and output layer
  - Input and output neurons are completely connected with each other
  - Suitable for
    - Regression of linear functions
    - Classification with linear discriminants

Multilayer Perceptron

Comparing MLP with SLP

- Now, we consider multilayer perceptron
  - Contains hidden layers between the input and output layers
  - The outputs of each layer are connected to inputs of all neurons of the respective next layer
  - Suitable for
    - Regression of all continuous functions with only one hidden layer
    - Regression also of non-continuous functions with at least two hidden layers
    - Classification with non-linear discriminants
  - Until 2007: In practice, one rarely uses more than one hidden layer, because the analysis of a network with many hidden layers turns out to be quite complicated

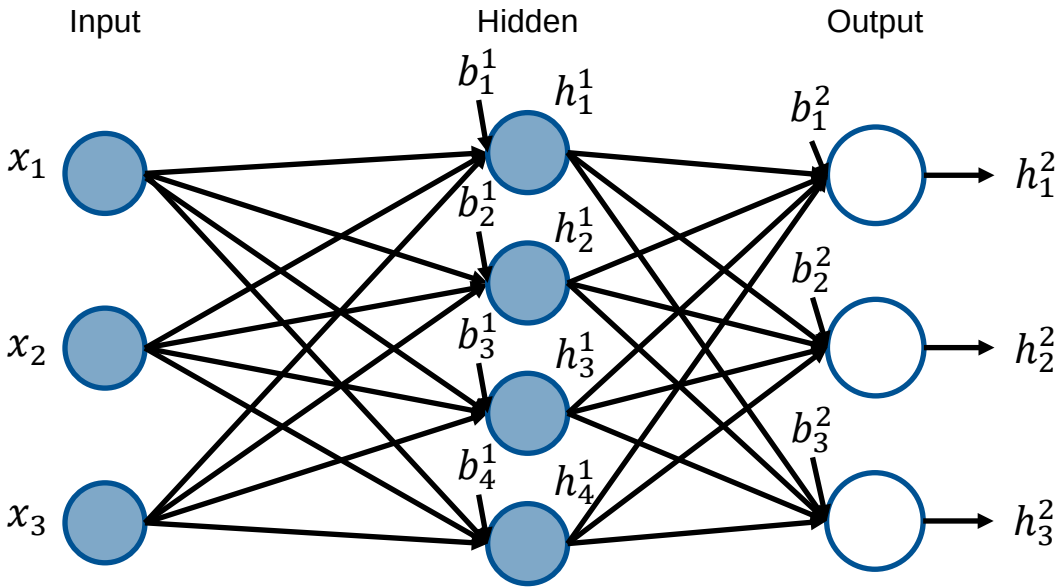
Multilayer Perceptron

Generalized framework

- An L-layer MLP consists of $L-1$ hidden layers and the output layer in depth L
- Input to the network: $\boldsymbol{h}^0 = \boldsymbol{x}$
- Output of the network: $\boldsymbol{h}^L = \boldsymbol{y}$
- Output of z_k neurons in layer k with $k \in (0, \dots, L)$: $\boldsymbol{h}^k = \left(h_1^k, h_2^k, \dots, h_{z_k}^k\right)$
- In the $(z_k \times z_{k-1})$-matrix $\boldsymbol{\Theta}^k$, the entry Θ_{ij}^k is the weight of the connection from the j-th neuron in layer $k-1$ to the i-th neuron in layer k
- The bias inputs to the $\boldsymbol{z}_k$ neurons in layer k are $\boldsymbol{b}_k = \left(b_1^k, b_2^k, \dots, b_{z_k}^k\right)$
- Calculation of the output in layer k for $k \in (1, \dots, L)$: $\boldsymbol{h}^k = f\left(\left(\boldsymbol{\Theta}^k\right)^{\mathrm{T}} \boldsymbol{h}^{k-1} + \boldsymbol{b}^k\right)$, where $f()$ denotes the activation function

Multilayer Perceptron

Topology of MLP with $K = 3$ and $L = 2$



Backpropagation

Basic principle of backpropagation learning algorithm

- Same approach as in the case of the **delta learning rule** (linear or logistic regression) for the SLP
- Formulate **error function $\boldsymbol{E}$** on the output of the MLP
- Determine dependence of the error function on the weights $E(\boldsymbol{\Theta})$
- Calculate partial derivatives $\frac{\partial E(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}}$ of the error function $E(\boldsymbol{\Theta})$
- Weight adjustment proportional to negative partial derivative

$$\Delta\Theta_{ij} = -\alpha \frac{\partial E(\boldsymbol{\Theta})}{\partial \Theta_{ij}}$$

- Separate consideration for weight adjustment of neurons
  - in the output layer
  - in the hidden layer

Backpropagation

Backpropagation of neurons of the output layer (regression)

- The output layer is a SLP taking the outputs of the last hidden layer as inputs
- In regression (supervised learning), the output y and the prediction can be used for the error function

$$E(\boldsymbol{\Theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(f\left(\left(\boldsymbol{\Theta}_i^L\right)^{\mathrm{T}} \cdot \boldsymbol{h}_i^{L-1} + b_i^L\right) - y_i\right)^2$$

- Weight adjustment as

$$\Delta\Theta_{ij} = -\alpha\frac{\partial E(\boldsymbol{\Theta})}{\partial\Theta_{ij}} = -\alpha\sum_{i=1}^{n}\left(f\left(\left(\boldsymbol{\Theta}_i^L\right)^{\mathrm{T}} \cdot \boldsymbol{h}_i^{L-1} + \boldsymbol{b}^L\right) - y\right) \cdot f' \cdot h_{ij}^{L-1} = \alpha\Delta_{ij} \cdot f' \cdot h_{ij}^{L-1}$$

- h_{ij}^{L-1} is the output of neuron j in layer $L-1$ for the i-training set

Backpropagation

Backpropagation of neurons of hidden layer (regression)

- In the hidden layers, the corresponding target output $\boldsymbol{h}_i^{L-1}$ is unknown
- In general, for the weight adjustment of the neurons in the hidden layer, the partial derivative of the error function is

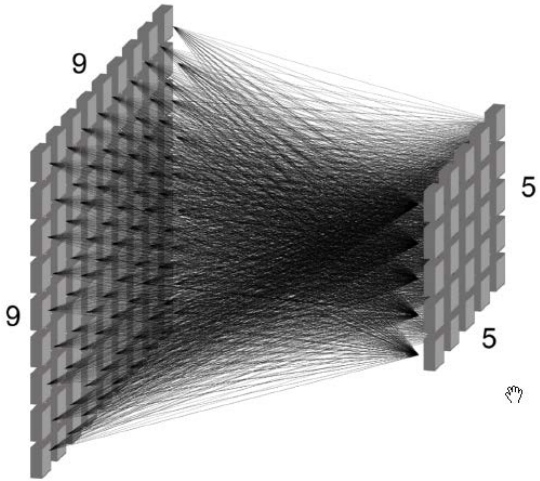
$$\frac{\partial E(\boldsymbol{\Theta})}{\partial \Theta_{ij}^{L-1}} = \frac{\partial E(\boldsymbol{\Theta})}{\partial f} \cdot \frac{\partial f}{\partial y_k} \cdot \frac{\partial y_k}{\partial \boldsymbol{h}_i^{L-1}} \cdot \frac{\partial \boldsymbol{h}_i^{L-1}}{\partial \Theta_{ij}^{L-1}}$$

- This way, the error contribution is back-propagated through the whole network to adjust weights
- For more information see:
  - https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
  - https://brilliant.org/wiki/backpropagation/#the-backpropagation-algorithm

Introduction

Motivation – When the models explode

- Lets assume a fully connected layer for image processing
  - Input is a 9×9, output is 5×5, with 4 Byte per parameter
  - Memory: $9 \cdot 9 \cdot 5 \cdot 5 \cdot 4$ Byte results in 8kB
  - FullHD to FullHD: $1920 \cdot 1080 \cdot 3 \cdot 1920 \cdot 1080 \cdot 3 \cdot 4\text{Byte} = 154\text{TB}$
- Number of required training data increases with number of parameters



Source: Adopted from TUM lecture: Prof. Lienkamp “Artificial Intelligence in Automotive Technology”

Frameworks

How to get started with Python

- Get Python distribution
  - Here we use Python 3.0 (e.g. Anaconda @ https://www.anaconda.com/products/distribution)
  - Install distribution on computer
  - Teach yourself (tutorials, youtube, TUM courses in Python)
- Understand structures of libraries
  - SciPy, NumPy, Scikit-Learn, Tensor Flow, PyTorch, matplotlib, Keras, mxnet, MS Deep Learning Toolkit, Caffe Deep Learning Lib Amazon DL Software DSSTNE, Chainer Neural Networks Lib, Theano Deep Learning Lib, PyMC

Source: https://www.anaconda.com/products/distribution

Links

Open data and further articles

- https://www.kdnuggets.com/ (Knowledge and database)
- https://www.kaggle.com/ (AI community and code samples)
- http://yann.lecun.com/exdb/mnist/ (Database for handwriting)

- https://doi.org/10.1121/1.5133944 ”Machine learning in acoustics: Theory and applications”
- https://doi:10.1038/nature14539 “Deep learning”, Review in Nature

Additional Courses

Further courses at TUM and Online on external platforms

- Coursera: Andrew Ng, Machine Learning, Stanford University
- Udacity
- Udemy
- Youtube
- Github, Google, etc.

Additional Literature

Books

- C. Bishop, Pattern Recognition and Machine Learning, New York: Springer, 2006.
- K. Murphy, Machine Learning: A Probabilistic Perspective, Cambridge: MIT Press, 2021.
- I. Goodfellow, Y. Bengio und A. Courville, Deep Learning, Frechen: Mitp-Verlag, 2016
- E. Alpaydin, Maschinelles Lernen, München: Oldenbourg Wissenschaftsverlag, 2008.
- G. Görz, J. Schneeberger und U. Schmid, Handbuch der Künstlichen Intelligenz, München: Oldenbourg Wissenschaftsverlag, 2014.
- S. Skansi, Introduction to Deep Learning, Berlin: Springer-Verlag, 2018.
- H. H. Aghdam und E. J. Heravi, Guide to Convolutional Neural Networks, Berlin: Springer-Verlag, 2017.

Machine Learning for Acoustics

- Supervised learning methods: discriminative models -

So far so good…

Thanks a lot!